c) Valores iniciais:

```
QUANTOS ← 0;
MAIORPULSOS ← IMPULSOS {deverá ser o primeiro valor lido  supondo que o
                        primeiro já é o maior}

leia (IMPULSOS);
MAIORPULSOS ← IMPULSOS;
```

passo 6: *Transformação 1*: "achar o maior IMPULSO e contar".

```
Para       ( IMPULSOS > MAIORPULSOS → {trocar o MAIORPULSO recomeçar a
cada       |                           contagem}
IMPULSOS   | IMPULSOS = MAIORPULSOS → {contar mais um}
lido       ( IMPULSOS < MAIORPULSOS → {nada a fazer}
```

```
se IMPULSOS > MAIORPULSOS
|  então MAIORPULSOS ← IMPULSOS; {trocar e recomeçar a contar}
|        QUANTOS ← QUANTOS + 1;
|  senão se IMPULSOS = MAIORPULSOS
|        |  então QUANTOS ← QUANTOS + 1; {contar}
|        fim se;
fim se;
```

passo 7: Supondo um cartão de dado com o valor 15, que MAIORPULSOS é igual a 10 e supondo QUANTOS igual a 8 teremos:

	MAIORIMPULSO	*IMPULSOS*	*QUANTOS*
antes →	10	15	8
depois →	15	15	1

Supondo agora que o cartão de dados possui também o valor 15, teremos:

	MAIORPULSOS	*IMPULSOS*	*QUANTOS*
antes :	15	15	1
depois :	15	15	2

finalmente, se o próximo cartão contiver o valor 13, teremos:

	MAIORPULSOS	*IMPULSOS*	*QUANTOS*
antes :	15	13	2
depois :	15	13	2

passo 8: a) *Considerações de integração*. O procedimento levantado em (6) deve ser repetido para cada assinante. O cartão com impulsos igual a − 1 não deve ser considerado; para isso, o comando repetitivo deve ser usado e o último comando dentro deste deve ser a leitura do novo valor.


```
enquanto IMPULSOS ≠ − 1 faça
|  "achar o maior impulso e contar";
|  leia (IMPULSOS);
fim enquanto;
```

b) *Elaboração do algoritmo:*

```
início {descobre o maior número de impulsos por assinante e indica quantos as-
|       sinantes atingiram este valor}
| inteiro: IMPULSOS,  {contém o número de impulsos por assinante}
|          QUANTOS,   {contém quantos assinantes atingiram o maior número de
|                      impulsos}
|          MAIORPULSOS; {conterá o maior número de impulsos encontrados}
| QUANTOS ← 0;
| leia (IMPULSOS);
|  MAIORPULSOS ← IMPULSOS; {supondo que o primeiro já é o maior}
| enquanto IMPULSOS ≠ − 1 faça
|    se IMPULSOS > MAIORPULSOS
|    |  então    MAIORPULSOS ← IMPULSOS;
|    |           QUANTOS ← 1;
|    |  senão se IMPULSOS = MAIORPULSOS
|    |          |  então QUANTOS ← QUANTOS + 1;
|    |          fim se;
|    fim se;
|    leia (IMPULSOS);
| fim enquanto;
| imprima ("MAIOR NÚMERO DE IMPULSOS NO MÊS:", MAIORPULSOS);
| imprima ("NÚMERO DE ASSINANTES:", QUANTOS);
fim.
```

À medida que o leitor vá adquirindo maior amadurecimento na confecção de algoritmos, vários dos passos descritos vão sendo feitos automaticamente. Recomenda-se que no início do aprendizado e quando se tratar de problemas mais complexos, os passos sejam seguidos formalmente.

No item seguinte serão apresentados diversos problemas com as respectivas soluções. Fica a cargo do leitor refazer, quando julgar necessário, os problemas segundo a metodologia apresentada, usando as especificações do problema e as soluções como referência.

3.3 PROBLEMAS PROPOSTOS E SOLUÇÕES

PROBLEMA 1

Escrever um algoritmo para gerar e imprimir N termos da série de Fibonacci:

1, 1, 2, 3, 5, 8, 13, 21, ...
N deve ser lido de cartão e $N > 2$